



下载APP



20 | 智能语音：好玩的语音控制是怎么实现的？

2020-12-25 郭朝斌

物联网开发实战

[进入课程 >](#)**讲述：郭朝斌**

时长 16:35 大小 15.20M



你好，我是郭朝斌。

实战篇的前几讲，我们打造了联网智能电灯，并实现了跟光照传感器的场景联动。今天我们来玩一个更酷的，智能音箱。

智能音箱为我们提供了一种更加自然的交互方式，所以亚马逊的 Echo 产品一经问世，就迅速流行起来。与智能家居结合之后，它更是引起了行业巨头的注意，被认为是很有发展潜力的用户入口和平台级产品。



我们先不论智能音箱最终到底能不能发展成智能家居的平台级产品，至少这波热潮已经极大地推动了相关技术的发展，而且用户覆盖率也有了很大的提升。

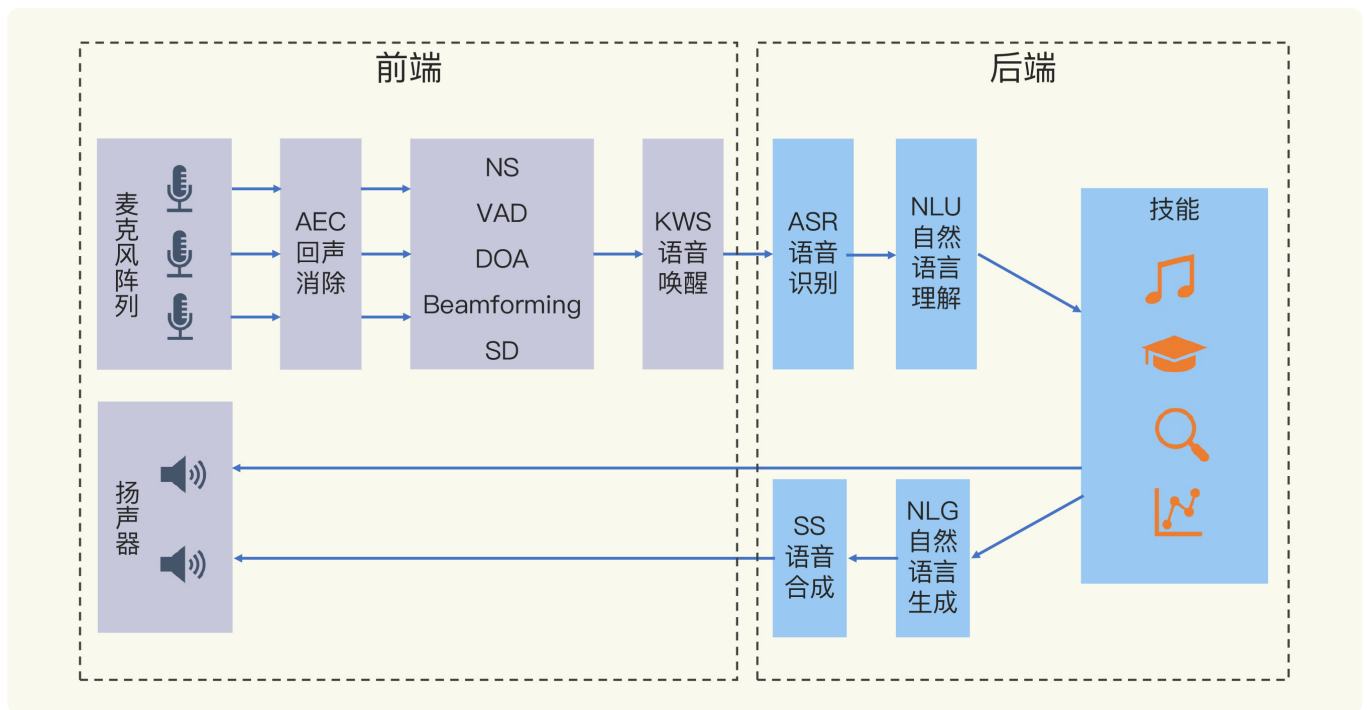
这一讲我就为你介绍一下智能音箱的语音控制是怎么实现的，并且带你动手完成开发过程。

智能音箱的技术架构

智能音箱主要涉及**拾音**、**前端信号处理**、**语音识别**、**自然语言处理**和**语音合成**等技术，现在一些产品甚至提供了声纹识别技术。

当然，智能音箱最重要的是提供各种功能，完成一些任务，比如控制电灯的开和关，这被称为**技能**。

整体的技术架构如下图所示：



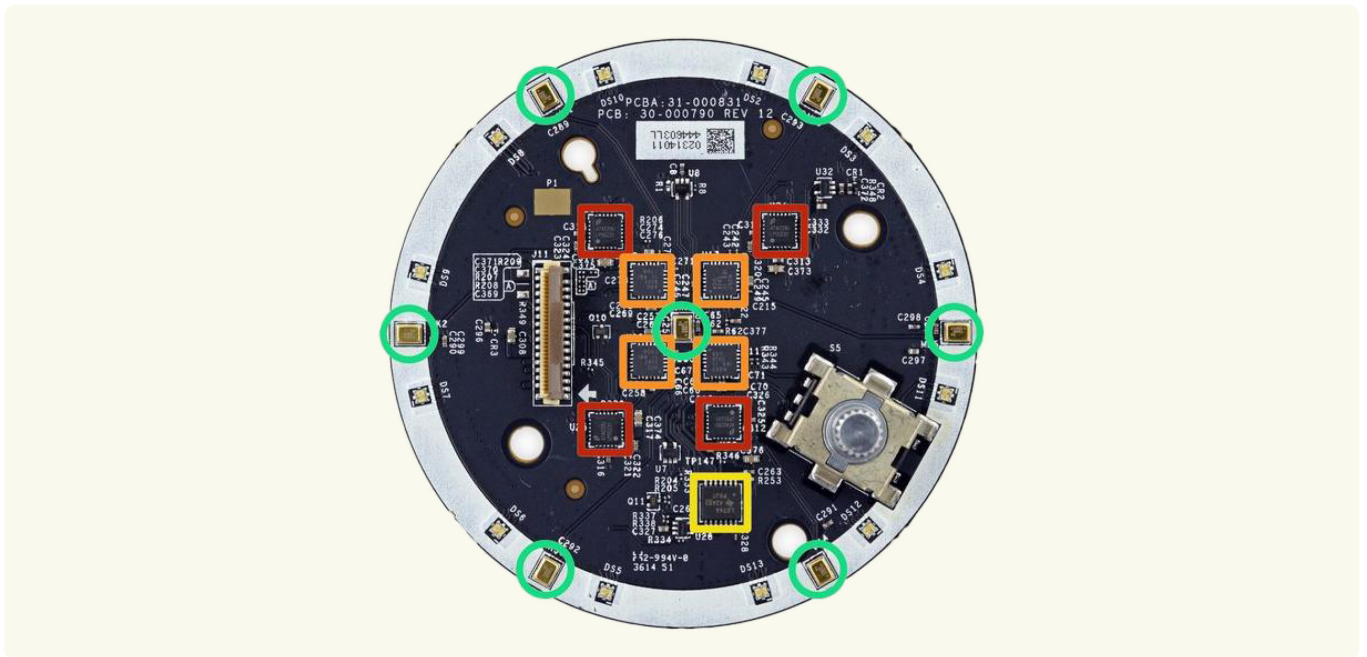
接下来，我会逐个讲解这些技术组成。

拾音

拾音，就是通过**麦克风**获取你的语音。

我们都用微信发送过语音消息，手机就是通过麦克风来获取你说的话的，这么说起来，拾音好像很简单。但是，智能音箱应对的环境要更复杂，因为用户可能在比较远的地方下达语音指令。

因此，智能音箱上一般采用**麦克风阵列**（Mic Array），也就是按照一定规则排列的多个麦克风，比如下图展示的就是 [Amazon Echo](#) 由 7 个麦克风组成的阵列（绿色圆圈部分）。



前端语音信号处理

在收集到声音信号后，还需要进行前端语音信号处理。只有经过处理，智能音箱才能获取到相对干净的语音信号，也才能提高后面的语音识别的准确率。

这些处理技术包括回声消除（Acoustic Echo Cancellation, AEC）、噪音抑制（Noise Suppression, NS）、语音检测（Voice Activity Detection, VAD）、声源定位（Direction of Arrival estimation, DOA）、波束成型（Beamforming）和混响消除（Speech Dereverberation）等。

回声消除（Acoustic Echo Cancellation, AEC），用于消除智能音箱在播放音乐时，外放的声音对麦克风采集声音的影响。

噪音抑制（Noise Suppression, NS），以便减低周围环境噪音的影响。

语音检测（Voice Activity Detection, VAD），主要是准确检测出语音的开始和结束，过滤掉非语音的声音。

声源定位（Direction of Arrival estimation, DOA），在麦克风阵列中，一般是基于声音到达的时间差等算法，确定声音源的空间位置。声源的空间位置可以帮助进行波束成型。

波束成型（Beamforming），在声音处理中是通过滤波算法，将声源方向的信号增强，而将其他方向的声音，比如其他方向传来的噪音，进行抑制，从而更好地实现降噪的目的。

混响消除（Speech Dereverberation, SD），通过算法将人声分离出来，为后续的语音唤醒和识别提供高质量的语音信号。

语音唤醒

语音唤醒 (Keyword Spotting, KWS) , 就是通过特定的**唤醒词**来激活智能音箱, 以便进行后续的语音交互任务。这样做一方面可以保护用户的隐私, 因为只有唤醒后, 音箱才收集和识别用户的语音信息, 另一方面也可以简化语音的识别和理解, 比如小米智能音箱的“小爱同学”就是这样的唤醒词。

语音识别

语音识别 (Automatic Speech Recognition, ASR) , 主要完成的任务是将语音转换成文本, 所以也被称为 STT (Speech to Text) 。

自然语言理解

自然语言理解 (Natural Language Understanding, NLU) , 是对语音识别生成的文本进行处理, 识别用户的意图, 并生产结构化的数据。

当然, 以现在的人工智能发展水平来看, 自然语言理解还有很长的路要走。这也是我们常发现智能音箱不够“智能”的原因。

技能

技能 (Skills) 一般要借助后端云平台的强大能力, 云平台可以提供知识图谱、家居设备远程控制 and 音乐等音频资源等能力。

自然语言生成

自然语言生成 (Natural Language Generation, NLG) , 就是将各种技能的响应结果组织成文本语言。比如当你询问天气时, 根据获取的天气状况和温度等信息生成“北京今天晴, 最高温度 5°, 最低温度 -6°”这样的语句。自然语言生成和自然语言理解都属于**自然语言处理** (Natural Language Processing, NLP) 的范畴。

语音合成

语音合成 (Speech Synthesis) , 就是将自然语言生成的文本转换为语音的形式, 提供给智能音箱播放出来, 给人的感觉就像和音箱在对话。因此, 这个过程也叫做 TTS (Text to Speech) 。

智能音箱的开发

了解完智能音箱的基本技术构成，下面我们就基于树莓派开发一个自己的简易智能音箱吧。

首先，我需要说明一下树莓派的系统。为什么呢？因为在 [第 15 讲](#) 中，我们安装了 Gladys Assistant 系统镜像，而这个系统 Raspbian 是基于 Debian buster 版本的，一些语音识别开源库对于 buster 的支持并不够好。

所以，如果你的树莓派是 Raspberry Pi 3 系列，强烈建议你把系统镜像切换成 **Debian stretch** 版本。通过 [这个链接](#) 就可以下载基于 Debian stretch 版本的 Raspbian 镜像文件压缩包，安装还是使用 Etcher 工具，你可以回头看一下第 15 讲的介绍。

至于树莓派 Raspberry Pi 4 系列，因为官方系统 Raspbian 只有 buster 版本支持，所以我们还是继续基于第 15 讲的系统开发。

麦克风阵列

麦克风阵列我使用的是 **ReSpeaker 2-Mics Pi HAT**，它的 2 个麦克风分布在模组的两边。我们现在来配置一下，让它可以在树莓派上正常工作。

你可以通过下面的命令安装它的驱动程序。首先，你最好切换一下树莓派的软件安装源，将它切换到国内的腾讯云安装源，这样下载安装的速度比较快。运行下面的命令修改配置文件：

```
1 $ sudo vim /etc/apt/sources.list
```

[复制代码](#)

将文件修改为下面的内容：

```
1 deb https://mirrors.cloud.tencent.com/raspbian/raspbian/ buster main contrib n
2 # Uncomment line below then 'apt-get update' to enable 'apt-get source'
3 deb-src https://mirrors.cloud.tencent.com/raspbian/raspbian/ buster main contr
```

[复制代码](#)

修改另一个软件安装源的配置文件，命令如下所示：

```
1 $ sudo vim /etc/apt/sources.list.d/raspi.list
```

[复制代码](#)

修改后的文件内容如下：

```
1 deb https://mirrors.cloud.tencent.com/raspberrypi/ buster main
2 # Uncomment line below then 'apt-get update' to enable 'apt-get source'
3 deb-src https://mirrors.cloud.tencent.com/raspberrypi/ buster main
```

[复制代码](#)

然后，你需要运行下面的命令更新安装源：

```
1 $ sudo apt-get clean all
2 $ sudo apt-get update
```

[复制代码](#)

现在，你可以运行下面命令安装麦克风阵列的驱动程序。因为这个驱动依赖的 wm8960 编解码器没有包含在树莓派系统的内核里面，需要重新加载内核，编译驱动，所以整个过程比较久。在等待的过程中，你可以先阅读这一讲的其他部分。

```
1 $ sudo apt-get install git
2 $ git clone --depth=1 https://github.com/respeaker/seeed-voicecard
3 $ cd seeed-voicecard
4 $ sudo ./install.sh
5 $ sudo reboot
```

[复制代码](#)

树莓派重启之后，你可以在树莓派终端输入下面的命令，查看音频的输入和输出设备是否正常工作。

```
1 $ arecord -l
2 $ aplay -l
```

[复制代码](#)

```
pi@gladys:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: seeed2micvoicec [seeed-2mic-voicecard], device 0: bcm2835-i2s-wm8960-hifi wm8960-hifi-0 [bcm2835-i2s-wm8960-hifi wm8960-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@gladys:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 Headphones]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: seeed2micvoicec [seeed-2mic-voicecard], device 0: bcm2835-i2s-wm8960-hifi wm8960-hifi-0 [bcm2835-i2s-wm8960-hifi wm8960-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

如果一切正常，我们就可以测试录音和播放功能了。在 ReSpeaker 2-Mics Pi HAT 的耳机插口上插入耳机或者扬声器，运行下面的命令，并说几句话。

[复制代码](#)

```
1 $ arecord -d 5 test.wav
2 $ aplay test.wav
```

另外，你也可以通过软件 **AlsaMixer**（命令 `alsamixer`）来配置声音设置和调整音量，左、右箭头键用于选择通道或设备，向上、向下箭头控制当前所选设备的音量。退出程序使用 `ALT + Q`，或者按 `Esc` 键。

为了简化开发，也考虑到麦克风硬件的限制，我们这里就先不关注前端语音信号处理的相关开发了。接下来，我们直接来到实现语音唤醒的环节。

语音唤醒

为了实现语音唤醒，我们需要选择一个轻量级的、可以在树莓派上运行的唤醒词监测器软件。

你可能首先想到的是 **Snowboy**，没错，它确实是一个非常流行的工具。不过，Snowboy 团队在 2020 年初的时候宣布，2020 年 12 月 31 日会停止提供服务，所以我们只能寻找替代方案。

我选择的是 [Mycroft Precise](#)，它是一个基于 RNN 神经网络的语音唤醒工具。

接下来，我们在树莓派安装 Mycroft Precise。因为需要训练唤醒词模型，我们需要基于源代码来编译、安装。

首先，我们通过 git 命令把 Mycroft Precise 的源代码下载到树莓派的 /home/pi 目录：

```
1 $ cd ~
2 $ git clone https://github.com/mycroftai/mycroft-precise
3 $ cd mycroft-precise
```

[复制代码](#)

在安装之前，把 pypi 的安装源修改到清华数据源，可以获得更快的下载速度。我们打开目录中的 setup.sh 文件：

```
1 $ vim setup.sh
```

[复制代码](#)

将文件中的这行内容：

```
1 extra-index-url=https://www.piwheels.org/simple
```

[复制代码](#)

替换成下面的内容：

```
1 index-url=https://pypi.tuna.tsinghua.edu.cn/simple
```

[复制代码](#)

```
2 extra-index-url=https://www.piwheels.org/simple
```

然后，我们运行它自带的安装脚本，开始编译和安装。中间如果执行中断，可以重新执行这个命令，继续安装过程。（提示：有些 ARM 平台的库只有 piwheels 上有，所以这些库安装时速度还是很慢。这种情况下，可以电脑上使用下载工具获取这个模块的安装文件，然后上传到树莓派上，手动安装。）

```
1 $ ./setup.sh
```

 复制代码

安装完成后，我们开始使用 Mycroft Precise 来训练一个唤醒词模型，唤醒词可以根据喜好来选择，比如“极客时间”。

我们需要先激活 Python 的虚拟环境，因为 Mycroft Precise 在安装过程中创建了这个虚拟环境。

```
1 $ source .venv/bin/activate
```

 复制代码

接下来，我们通过工具 precise-collect 来收集语音模型训练的声音素材，运行后，根据提示录制 12 段声音。

```
1 $ precise-collect
2 Audio name (Ex. recording-##): geektime.##
3
4 Press space to record (esc to exit)...
5 Recording...
6 Saved as geektime-00.wav
7 Press space to record (esc to exit)...
```

 复制代码

然后，我们需要将这些声音随机分为两份，一份是训练样本，包括 8 个声音文件，另一份是测试样本，包括 4 个声音文件，并且把这两份样本分别放到 geektime/wake-word/ 和 /geektime/test/wake-word/ 这两个目录下。

接着，我们执行下面的命令，生成神经网络模型 `geektime.net`：

```
1 $ precise-train -e 60 geektime.net geektime/
```

[复制代码](#)

最后，我们还需要将 `geektime.net` 的模型格式做一下转换，将它从 Keras 模型格式改为 TensorFlow 模型格式，因为 TensorFlow 模型更加通用。

```
1 $ precise-convert geektime.net
```

[复制代码](#)

执行完成之后，我们会得到两个文件：

1. `geektime.pb`, TensorFlow 模型文件
2. `geektime.pb.params`, 包含 Mycroft Precise 在处理音频时需要的一些参数信息。

当然，为了提高模型的准确性，我们还可以使用 `precise-train-incremental` 工具来增加负样本，重新训练刚才的模型。如果环境复杂的话，你可以尝试一下。

然后，我们可以运行一段代码来测试这个唤醒词模型。不过，因为 `portaudio` 这个库在树莓派上运行有问题，我们需要先修复一下 `portaudio` 库。你可以运行下面的命令：

```
1 $ sudo apt-get remove libportaudio2
2 $ sudo apt-get install libasound2-dev
3 $ git clone -b alsapatch https://github.com/gglockner/portaudio
4 $ cd portaudio
5 $ ./configure && make
6 $ sudo make install
7 $ sudo ldconfig
```

[复制代码](#)

测试程序的代码如下：

```
1 # File: kwsdemo.py
```

[复制代码](#)

```
2 #!/usr/bin/env python3
3
4 from precise_runner import PreciseEngine, PreciseRunner
5
6 engine = PreciseEngine('precise-engine/precise-engine', 'geektime.pb')
7 runner = PreciseRunner(engine, on_activation=lambda: print('hello'))
8 runner.start()
9
10 # Sleep forever
11 from time import sleep
12 while True:
13     sleep(10)
```

现在，我们把 kwsdemo.py 文件，还有两个 geektime.pb 模型相关的文件，都上传到树莓派的 Mycroft Precise 目录下，然后运行 kwsdemo.py 文件，说出“极客时间”几个字，就会看到终端显示出“hello”这个单词。

语音识别

对于语音识别，我们直接采用腾讯云提供的语音识别 SDK 来完成（你需要提前在腾讯云控制台开通这个服务）。它会将语音发送到云端，由云端服务器计算出文本信息。你可以通过下面命令来安装：

```
1 $ pip3 install tencentcloud-sdk-python
```

[📄 复制代码](#)

在开始使用之前，你需要访问 [这个链接](#) 创建一个密钥，然后记录下 SecretId 和 SecretKey 的信息。

你可以参考下面的代码，来完成一个录音文件的识别。

```
1 from tencentcloud.common import credential
2 from tencentcloud.common.profile.client_profile import ClientProfile
3 from tencentcloud.common.profile.http_profile import HttpProfile
4 from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentC
5 from tencentcloud.asr.v20190614 import asr_client, models
6 import base64
7 import io
8 import sys
```

[📄 复制代码](#)

```
9 SECRET_ID = "你的Secret ID"
10 SECRET_KEY = "你的Secret Key"
11
12 try:
13     cred = credential.Credential(SECRET_ID, SECRET_KEY)
14     httpProfile = HttpProfile()
15     httpProfile.endpoint = "asr.tencentcloudapi.com"
16     clientProfile = ClientProfile()
17     clientProfile.httpProfile = httpProfile
18     clientProfile.signMethod = "TC3-HMAC-SHA256"
19     client = asr_client.AsrClient(cred, "ap-beijing", clientProfile)
20     #读取文件以及 base64
21     with open('./geektime-00.wav', "rb") as f:
22         if sys.version_info[0] == 2:
23             content = base64.b64encode(f.read())
24         else:
25             content = base64.b64encode(f.read()).decode('utf-8')
26         f.close()
27     #发送请求
28     req = models.SentenceRecognitionRequest()
29     params = {"ProjectId":0,"SubServiceType":2,"SourceType":1,"UsrAudioKey":"s
30     req._deserialize(params)
31     req.DataLen = len(content)
32     req.Data = content
33     req.EngSerViceType = "16k_zh"
34     req.VoiceFormat = "wav"
35     resp = client.SentenceRecognition(req)
36     print(resp.to_json_string())
37
38 except TencentCloudSDKException as err:
39     print(err)
40
```

语音合成

接下来，我来介绍一下语音合成。

你可能会问，刚才介绍技术架构的时候，不是还讲了自然语言理解、技能和自然语言生成吗？这里怎么跳过去了呢？

首先，因为我们的任务很简单，只需要查询语音识别的文本中是否有“开”、“灯”，和“关”、“灯”就可以完成判断，所以自然语言理解直接判断字符串是否匹配即可。

其次，我们要实现控制智能电灯，这个技能我在后面会介绍。

最后，智能音箱只需要反馈执行开关灯的结果就可以，比如“我已经把灯打开了”或者“我已经把灯关了”，自然语言生成的部分按照固定的文本就可以了，不需要考虑动态生成的问题。

语音合成，就是我们希望把类似“我已经把灯关了”这样的文本信息，转换为音频，便于智能音箱播放出来。你可以基于离线的 TTS 引擎来实现，比如 [🔗 HanTTS](#) 这个项目。

当然，我们也可以使用腾讯云的语音合成服务（你需要提前在腾讯云控制台开通这个服务）。你可以参考下面的代码：

[📄 复制代码](#)

```
1 import json
2 import base64
3
4 from tencentcloud.common import credential
5 from tencentcloud.common.profile.client_profile import ClientProfile
6 from tencentcloud.common.profile.http_profile import HttpProfile
7 from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentC
8 from tencentcloud.tts.v20190823 import tts_client, models
9
10 SECRET_ID = "你的Secret ID"
11 SECRET_KEY = "你的Secret Key"
12
13 try:
14     cred = credential.Credential(SECRET_ID, SECRET_KEY)
15     httpProfile = HttpProfile()
16     httpProfile.endpoint = "tts.tencentcloudapi.com"
17
18     clientProfile = ClientProfile()
19     clientProfile.httpProfile = httpProfile
20     client = tts_client.TtsClient(cred, "ap-beijing", clientProfile)
21
22     req = models.TextToVoiceRequest()
23     params = {
24         "Text": "我已经把灯关了",
25         "SessionId": "sessionid-geektime",
26         "ModelType": 1,
27         "ProjectId": 0,
28         "VoiceType": 1002
29     }
30     req.from_json_string(json.dumps(params))
31
32     resp = client.TextToVoice(req)
33     print(resp.to_json_string())
34
35     if resp.Audio is not None:
```

```
36     audio = resp.Audio
37     data = base64.b64decode(audio)
38     wav_file = open("temp.wav", "wb")
39     wav_file.write(data)
40     wav_file.close()
41
42 except TencentCloudSDKException as err:
43     print(err)
```

通过智能音箱控制电灯

为了实现控制智能电灯的目的，我们需要借助物联网平台提供的开发接口。

首先，我们进入物联网开发平台，选择“智能家居”项目。



然后，点击左侧的“应用开发”，进入新建应用的界面，点击“新建应用”。

完成后，点击应用列表里面的应用名称，进入应用的详情页面。你可以看到应用的 SecretId 和 SecretKey 信息。这里，你需要将下面“关联产品”中的智能电灯勾选上。只有建立关联，应用才可以控制这个设备。

具体代码可以参考腾讯提供的开源实现，包括 [iOS](#)、[Android](#) 和 [小程序](#)。

不过，这种方式需要用户账号的登录认证，在树莓派上不太方便。还有一个方式就是基于物联网开发平台提供的通用 API 接口。其中的“[设备远程控制](#)”接口可以满足我们的需求。

具体的控制方法，你可以参考下面的代码（注意，目前只支持 ap-guangzhou 区域）。

```
1 import json
2 from led2.main import PRODUCT_ID
3 from tencentcloud.common import credential
4 from tencentcloud.common.profile.client_profile import ClientProfile
5 from tencentcloud.common.profile.http_profile import HttpProfile
6 from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentC
7 from tencentcloud.iotexplorer.v20190423 import iotexplorer_client, models
8
9 SECRET_ID = "你的Secret ID"
10 SECRET_KEY = "你的Secret Key"
11 PRODUCT_ID = "你的ProductID"
12
13 def Light_control(state):
14     try:
15         cred = credential.Credential(SECRET_ID, SECRET_KEY)
16         httpProfile = HttpProfile()
17         httpProfile.endpoint = "iotexplorer.tencentcloudapi.com"
18
19         clientProfile = ClientProfile()
20         clientProfile.httpProfile = httpProfile
21         client = iotexplorer_client.IotexplorerClient(cred, "ap-guangzhou", cl
22
23         req = models.ControlDeviceDataRequest()
24         data = {
25             "power_switch": state
26         }
27         data_str = json.dumps(data)
28
29         params = {
30             "DeviceName": "Led_1",
31             "ProductId": PRODUCT_ID,
32             "Data": data_str
33         }
34         req.from_json_string(json.dumps(params))
35
36         resp = client.ControlDeviceData(req)
37         print(resp.to_json_string())
38
39     except TencentCloudSDKException as err:
40         print(err)
41
42 Light_control(0)
```

小结

总结一下，在这一讲中，我介绍了智能音箱的技术架构，以及在树莓派上用于实现智能音箱的一些可选的技术方案，并且带你实现了语音控制智能电灯的目的。你需要重点关注的

知识有：

1. 智能音箱的实现，需要前端音箱本体和后端云平台上一系列技术的支持。这些技术有前端的拾音、语音信号处理、语音唤醒和播音，以及后端的语音识别、自然语言理解、技能、自然语言生成和语音合成。
2. 在树莓派的实现上，拾音可以选择使用麦克风阵列，因为基于麦克风阵列可以更好地实现前端语音信号处理，比如声源定位和波束成型等。
3. 语言唤醒需要在智能音箱本体上实现，所以需要一些轻量级的识别引擎和训练好的唤醒词模型。之前比较流行的 Snowboy 将要停止服务，这里我选择了 Mycroft Precise 这个开源方案。
4. 语音识别、自然语言理解、技能、自然语言生成和语音合成等任务适合基于云平台的能力来实现，因为云平台的计算能力更强，有更好的性能和准确度。

智能音箱的技术也一直在发展，比如现在越来越多的智能音箱开始配备屏幕和摄像头，这为智能音箱引入了声音、UI 和视觉等多模态的交互方式，相应地，这也给声纹识别、人脸识别和动作识别等技术带来了新的应用场景。我相信智能音箱未来的产品形态和功能还会不断地进化和发展。

思考题

最后，我给你留一个思考题吧。

在这一讲中，我们是通过物联网平台提供的 API 接口来控制智能电灯的。除了这种方式，你还能想到其他的方法来远程控制智能电灯吗？你能实现一个虚拟的联网开关，基于场景联动来控制智能电灯的开和关吗？

欢迎你在留言区写下你思考的结果，也欢迎你将这一讲分享给你的朋友，大家一起交流学习。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 19 | 场景联动：智能电灯如何感知光线？（下）

下一篇 21 | 多传感器集成：浇花怎么实现自动化？

精选留言 (7)

写留言



信大捷安 置顶

2021-01-17

腾讯镜像经常会出现同步失败的问题，如果出现不能正常下载安装的情况可以使用清华大学镜像源

编辑 `/etc/apt/sources.list` 文件，删除原文件所有内容，用以下内容取代：
`deb http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main non-fre...`

展开



孙宇翔

2020-12-25

我想等把课程里的都实现后，试着在门框的两边装两个红外传感器，通过感知的先后顺序，来判断是进门或者出门，来控制灯的开关

展开

作者回复：好，等分享



2

6



莹

2021-01-03

使用precise-convert geektime.net会出现一点问题，说是有些文件中的`.decode('utf8')`有AttributeError，估计是版本问题，看到最近有个报了的bug：<https://github.com/MycroftAI/mycroft-precise/issues/197>，暂时的解决办法，将报错中的`.decode('utf8')`删掉

展开

作者回复：👍





3

**莹**

2021-01-03

上面的kwsdemo.py的测试代码里这第6行"precise-engine/precise-engine"会导致找不到文件，需要替换到 ".venv/bin/precise-engine "。参考github上的sample python program.

作者回复: 记得执行时切换到precise的python虚拟环境



2

**牛哥哥**

2021-01-09

arecord -l 发现没有设备，重装系统也不行

展开 ∨

作者回复: 需要再看看respeaker的驱动安装是否成功



4

1

**Geek_e3bbee**

2021-01-12

语言唤醒 部分能跑起来，但最终唤醒效果没有出现，不知道其他同学有没有碰到类似的问题

作者回复: 可以关注负样本，增加训练模型的质量。



2

**LDxy**

2020-12-25

文中提到的智能音箱的技术架构中所用的各种技术，是不是都已经有了成熟的解决方案的？并且都已经能在产品中使用的？

展开 ∨

作者回复: 是的，都比较成熟了



